eip ESCUELA INTERNACIONAL POSGRADOS

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 10

Servicios de red y aplicaciones web

ÍNDICE

INTRODUCCIÓN

En este último capítulo veremos una pequeña introducción a los servicios de red donde hablaremos de los sockets y cómo trabajar con ellos.

Por otro lado, también veremos cómo crear un sitio web dinámico con Python

OBJETIVOS

Al finalizar esta lección serás capaz de:

1. Trabajar con socket
2. Crear un entorno virtual
3. Crear una aplicación web
4. Conocer otros framework de desarrollo web

PROGRAMACIÓN EN RED

Python realiza servicios de red mediante dos niveles de acceso. Nivel bajo y nivel superior:

Nivel bajo:
Puede acceder al soporte básico de sockets en el sistema operativo subyacente que implementa clientes y servidores para protocolos orientados a la conexión y sin conexión.

Nivel superior:
Puede acceder a través de librerías Python a protocolos de red específicos de nivel de aplicación como FTP, HTTP, etc.

Sockets

Los puntos finales de un canal de comunicación bidireccional son sockets. Los sockets pueden comunicar procesos entre sí utilizando la misma o distinta mÁquina. Estos pueden ser implementados usando una variedad de tipos de canales, incluyendo sockets de dominio Unix, TCP y UDP.

Módulo socket

El módulo socket se utiliza para implementar los sockets a través de diferentes canales. La función socket () es utiliza para crear un socket.

Sintaxis:

socket. socket()

Servidor simple

Un programa de servidor tiene los siguientes puntos:

- Importar el módulo socket
- Crear un objeto socket mediante socket ()
- Obtener el nombre de la máquina local mediante gethostname ()
- Reservar un puerto para su servicio.
- Enlaza con el puerto usando bind ()
- Luego espera la conexión del cliente
- Establezca la conexión con el cliente llegado.
- Por último cerrar la conexión

Servidor simple

```
# Import socket module
import socket

# Create a socket object
s = socket.socket()
# Get local machine name
host = socket.gethostname()
# Reserve a port for your service
port = 12345
```

```
# Bind to the port
s.bind((host, port))
# Now wait for client connection
s.listen(5)

while True:
   # Establish connection with client
   c, addr = s.accept()
   print('Got connection from', addr)
   c.send('Thank you for connecting')
   # Close the connection
   c.close()
```

Cliente simple

Una vez que el socket se abre, podemos leer de él. Tendremos que cerrar el socket una vez finalizado. El cliente debe tener los siguientes pasos:

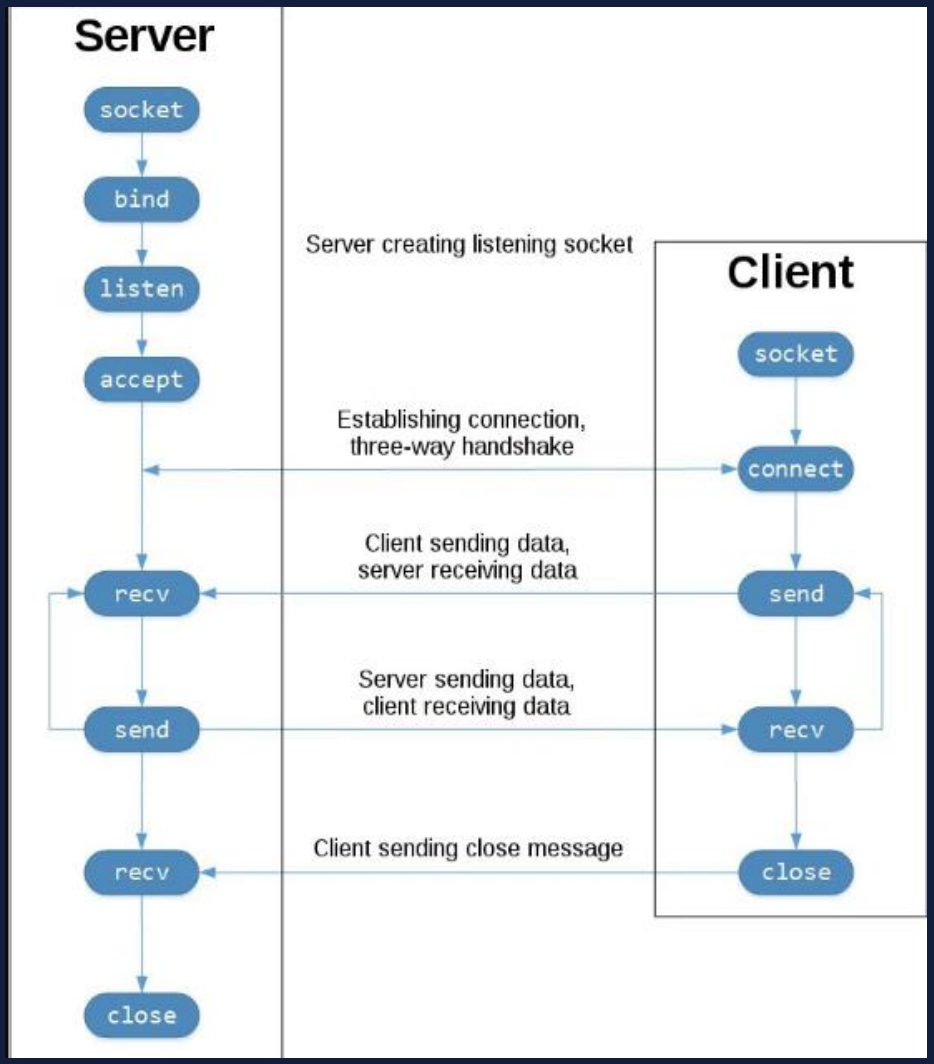
- Importar el módulo socket
- Crear un objeto socket
- Obtener el nombre de la máquina local
- Reservar un puerto para el servicio
- Abre una conexión tcp al nombre de la máquina en el puerto
- Recibe los mensajes tcp
- Cierra el socket al final

Cliente simple

```
# Import socket module
import socket

 # Create a socket object
s = socket.socket()
# Get local machine name
host = socket.gethostname()
 # Reserve a port for your service.
port = 12345

# open a tcp conncetion
s.connect((host, port))
# receive tcp messages
print(s.recv(1024))
# Close the socket when done
s.close()
```

```
# Following would start a server in background.
$ python server.py &

# Once server is started run client as follows:
$ python client.py
```

```
Got connection from ('127.0.0.1', 48437)
Thank you for connecting
```


Server
socket
bind
listen
accept
recv
send
recv
close
Server creating listening socket
Establishing connection,
three-way handshake
Client sending data,
server receiving data
Server sending data,
client receiving data
Client sending close message
Client
socket
connect
send
recv
close

APLICACIONES WEB

Una aplicación web es una de las mejores y tiene la ventaja de ser independiente de la plataforma. Por lo tanto, puede ser ejecutada por cualquier persona que utilice Internet. Hay diferentes tipos de aplicaciones web:

- **Aplicaciones web estáticas:**

  En este tipo los sitios web tienen contenidos fijos y su contenido no cambia cuando se interactúa con él. No se consideran aplicaciones porque no son dinámicas.

- **Aplicaciones web dinámicas:**

  Las aplicaciones web dinámicas son consideradas como verdaderas porque actúan dinámicamente y cambian su contenido al interactuar con ella. La aplicación de correo web es uno de los ejemplos que permiten al usuario interactuar con ella de muchas maneras y recibimos los correos a medida que van llegando.

Construir una aplicación web básica

Configurar el proyecto

Creamos una carpeta de proyecto y le asignamos un nombre que sea descriptivo de nuestro proyecto. Por ejemplo llamemos a la carpeta hello-app. Necesitaremos dos archivos dentro de esta carpeta:

- main.py contiene tu código Python envuelto en una implementación mínima del framework web Flask.
- requirements.txt lista todas las dependencias que tu código necesita para funcionar correctamente.

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Congratulations, it's a web app!"
```

```
Flask==1.1.2
```

CONCLUSIONES

1. Un socket es un tipo de punto final que funciona para establecer un enlace de red bidireccional entre un servidor y un cliente.

2. **Virtualenv** es una herramienta para crear entornos Python aislados.

3. **Flask** es un marco de aplicación web ligero . Está diseñado para que la puesta en marcha sea rápida y sencilla, con la capacidad de escalar a aplicaciones complejas.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

rsanchezi@grupomainjobs.com

Rubén Sánchez Iruela
linkedin.com/in/ruben-sanchez-iruela-8156799a

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados